

Further Algebraic Algorithms in the Congested Clique Model and Applications to Graph-Theoretic Problems

François Le Gall
Graduate School of Informatics
Kyoto University
legall@i.kyoto-u.ac.jp

Abstract

Censor-Hillel et al. [PODC'15] recently showed how to efficiently implement centralized algebraic algorithms for matrix multiplication in the congested clique model, a model of distributed computing that has received increasing attention in the past few years. This paper develops further algebraic techniques for designing algorithms in this model. We present deterministic and randomized algorithms, in the congested clique model, for efficiently computing multiple independent instances of matrix products, computing the determinant, the rank and the inverse of a matrix, and solving systems of linear equations. As applications of these techniques, we obtain more efficient algorithms for the computation, again in the congested clique model, of the all-pairs shortest paths and the diameter in directed and undirected graphs with small weights, improving over Censor-Hillel et al.'s work. We also obtain algorithms for several other graph-theoretic problems such as computing the number of edges in a maximum matching and the Gallai-Edmonds decomposition of a simple graph, and computing a minimum vertex cover of a bipartite graph.

1 Introduction

Background. The congested clique model is a model in distributed computing that has recently received increasing attention [5, 12, 13, 17, 18, 19, 20, 26, 27, 30, 34, 35]. In this model n nodes communicate with each other over a fully-connected network (i.e., a clique) by exchanging messages of size $O(\log n)$ in synchronous rounds. Compared with the more traditional congested model [36], the congested clique model removes the effect of distances in the computation and thus focuses solely on understanding the role of congestion in distributed computing.

Typical computational tasks studied in the congested clique model are graph-theoretic problems [5, 12, 13, 17, 20, 34], where a graph G on n vertices is initially distributed among the n nodes of the network (the ℓ -th node of the network knows the set of vertices adjacent to the ℓ -th vertex of the graph, and the weights of the corresponding edges if the graph is weighted) and the nodes want to compute properties of G . Besides their theoretical interest and potential applications, such problems have the following natural interpretation in the congested clique model: the graph G represents the actual topology of the network, each node knows only its neighbors but can communicate to all the nodes of the network, and the nodes want to learn information about the topology of the network.

Censor-Hillel et al. [5] recently developed algorithms for several graph-theoretic problems in the congested clique model by showing how to implement centralized algebraic algorithms for matrix multiplication in this model. More precisely, they constructed a $O(n^{1-2/\omega})$ -round algorithm for matrix multiplication, where ω denotes the exponent of matrix multiplication (the best known upper bound on ω is $\omega < 2.3729$, obtained in [25, 43], which gives exponent $1 - 2/\omega < 0.1572$ in the congested clique model), improving over the $O(n^{2-\omega})$ algorithm mentioned in [13], in the following setting: given two $n \times n$ matrices A and B over a field, the ℓ -th node of the network initially owns the ℓ -th row of A and the ℓ -column of B , and needs to output the ℓ -th row and the ℓ -column of the product AB . Censor-Hillel et al. consequently obtained $O(n^{1-2/\omega})$ -round algorithms for several graph-theoretic tasks that reduce to computing the powers of (some variant of) the adjacency matrix of the graph, such as counting the number of triangles in a graph (which lead to an improvement over the prior best algorithms for this task [12, 13]), detecting the existence of a constant-length cycle and approximating the all-pairs shortest paths in the input graph (improving the round complexity obtained in [34]). One of the main advantages of such an algebraic approach in the congested clique model is its versatility: it makes possible to construct fast algorithms for graph-theoretic problems, and especially for problems for which the best non-algebraic centralized algorithm is highly sequential and does not seem to be implementable efficiently in the congested clique model, simply by showing a reduction to matrix multiplication (and naturally also showing that this reduction can be implemented efficiently in the congested clique model).

Our results. In this paper we develop additional algebraic tools for the congested clique model.

We first consider the task of computing in the congested clique model not only one matrix product, but multiple independent matrix products. More precisely, given k matrices A_1, \dots, A_k each of size $n \times m$ and k matrices B_1, \dots, B_k each of size $m \times m$, initially evenly distributed among the n nodes of the network, the nodes want to compute the k matrix products $A_1 B_1, \dots, A_k B_k$. Prior works [5, 13] considered only the case $k = 1$ and $m = n$, i.e., one product of two square matrices. Our contribution is thus twofold: we consider the rectangular case, and the case of several matrix products as well. Let us first discuss our results for square matrices ($m = n$). By using sequentially k times the matrix multiplication algorithm from [5], k matrix products can naturally be computed in $O(kn^{1-2/\omega})$ rounds. In this work we show that we can actually do better.

Theorem 1 (Simplified version). *In the congested clique model k independent products of pairs of $n \times n$ matrices can be computed with round complexity*

$$\begin{cases} O(k^{2/\omega} n^{1-2/\omega}) & \text{if } 1 \leq k < n, \\ O(k) & \text{if } k \geq n. \end{cases}$$

This generalization of the results from [5] follows from a simple strategy: divide the n nodes of the network into k blocks (when $k \leq n$), each containing roughly n/k nodes, compute one of the k matrix products per block by using an approach similar to [5] (i.e., a distributed version of the best centralized algorithm computing one instance of square matrix multiplication), and finally distribute the relevant part of the k output matrices to all the nodes of the network. Analyzing the resulting protocol shows that the dependence in k in the overall round complexity is reduced to $k^{2/\omega}$. This sublinear dependence in k has a significant number of implications (see below).

The complete version of Theorem 1, given in Section 3, also considers the general case where the matrices may not be square (i.e., the case $m \neq n$), which will be crucial for some of our applications to the All-Pairs Shortest Path problem. The proof becomes more technical than for the square case, but is conceptually very similar: the main modification is simply to now implement a distributed version of the best centralized algorithm for rectangular matrix multiplication. The upper bounds obtained on the round complexity depend on the complexity of the best centralized algorithms for rectangular matrix multiplication (in particular the upper bounds given in [24]). Figure 1 depicts the upper bounds we obtain for the case $k = 1$. While the major open problem is still whether the product of two square matrices can be computed in a constant (or nearly constant) number of rounds, our results show that for $m = O(n^{0.651\dots})$, the product of an $n \times m$ matrix by an $m \times n$ matrix can indeed be computed in $O(n^\epsilon)$ rounds for any $\epsilon > 0$. We also show lower bounds on the round complexity of the general case (Proposition 1 in Section 3), which are tight for most values of k and m , based on simple arguments from communication complexity.

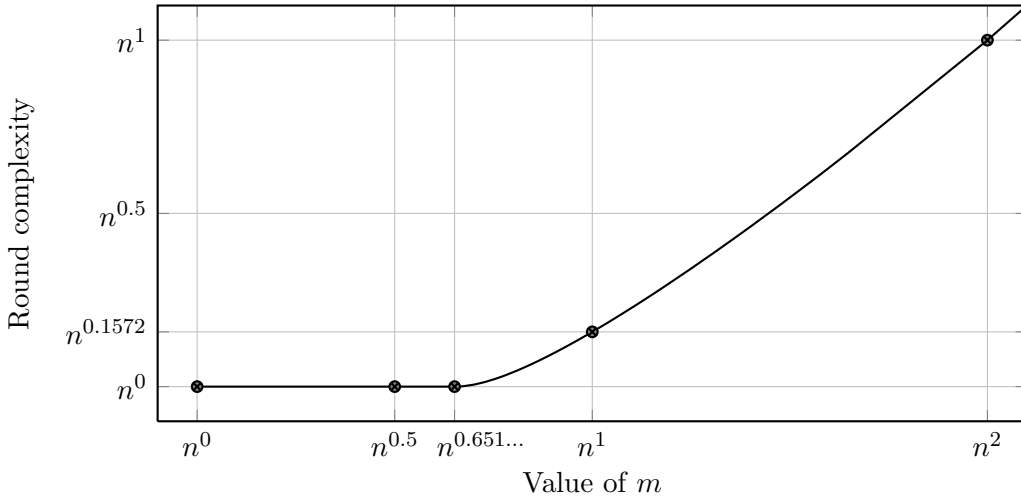


Figure 1: Our upper bounds on the round complexity of the computation of the product of an $n \times m$ matrix by an $m \times n$ matrix in the congested clique model.

We then study the following basic problems in linear algebra: computing the determinant, the rank or the inverse of an $n \times n$ matrix over a finite field \mathbb{F} of order upper bounded by a polynomial of n , and solving a system of n linear equations and n variables. We call these problems $\text{DET}(n, \mathbb{F})$,

$\text{Rank}(n, \mathbb{F})$, $\text{INV}(n, \mathbb{F})$ and $\text{SYS}(n, \mathbb{F})$, respectively (the formal definitions are given in Section 2). While it is known that in the centralized setting these problems can be solved with essentially the same time complexity as matrix multiplication [4], these reductions are typically sequential and do not work in a parallel setting. In this paper we design fast deterministic and randomized algorithm for these four basis tasks, and obtain the following results.

Theorem 2. *Assume that \mathbb{F} has characteristic greater than n . In the congested clique model, the deterministic round complexity of $\text{DET}(n, \mathbb{F})$ and $\text{INV}(n, \mathbb{F})$ is $O(n^{1-1/\omega})$.*

Theorem 3. *Assume that \mathbb{F} has order $|\mathbb{F}| = \Omega(n^2 \log n)$. In the congested clique model, the randomized round complexity of $\text{DET}(n, \mathbb{F})$, $\text{SYS}(n, \mathbb{F})$ and $\text{Rank}(n, \mathbb{F})$ is $O(n^{1-2/\omega} \log n)$.*

The upper bounds of Theorems 2 and 3 are $O(n^{0.5786})$ and $O(n^{0.1572})$, respectively, by basing our implementation on the asymptotically fastest (but impractical) centralized algorithm for matrix multiplication corresponding to the upper bound $\omega < 2.3729$. These bounds are $O(n^{2/3})$ and $O(n^{1/3} \log n)$, respectively, by basing our implementation on the trivial (but practical) centralized algorithm for matrix multiplication (corresponding to the bound $\omega \leq 3$). These algorithms are obtained by carefully adapting to the congested clique model the relevant known parallel algorithms [9, 21, 22, 23, 37] for linear algebra, and using our efficient algorithm for computing multiple matrix products (Theorem 1) as a subroutine. An interesting open question is whether $\text{INV}(n, \mathbb{F})$ can be solved with the same (randomized) round complexity as the other tasks. This problem may very well be more difficult; in the parallel setting in particular, to the best of our knowledge, whether matrix inversion can be done with the same complexity as these other tasks is also an open problem.

Applications of our results. The above results give new algorithms for many graph-theoretic problems in the congested clique model, as described below and summarized in Table 1.

Our main key tool to derive these applications is Theorem 7 in Section 3, which gives an algorithm computing efficiently the distance product (defined in Section 2) of two matrices with small integer entries based on our algorithm for multiple matrix multiplication of Theorem 1. Computing the distance product is a fundamental graph-theoretic task deeply related to the All-Pairs Shortest Path (APSP) problem [40, 41, 47]. Combining this result with techniques from [41], and observing that these techniques can be implemented efficiently in the congested clique model, we then almost immediately obtain the following result.

Theorem 4. *In the congested clique model, the deterministic round complexity of the all-pairs shortest paths problem in an undirected graph of n vertices with integer weights in $\{0, \dots, M\}$, where M is an integer such that $M \leq n$, is $\tilde{O}(M^{2/\omega} n^{1-2/\omega})$.*

Since computing the diameter of a graph reduces to solving the all-pairs shortest paths, we obtain the same round complexity for diameter computation in the same class of graphs. This improves over the $\tilde{O}(Mn^{1-2/\omega})$ -round algorithm for these tasks (implicitly) given in [5]. The main application of our results nevertheless concerns the all-pair shortest paths problem over directed graphs (for which the approach based on [41] does not work) with constant weights. We obtain the following result by combining our algorithm for distance product computation with Zwick's approach [47].

Theorem 5. *In the congested clique model, the randomized round complexity of the all-pairs shortest paths problem in a directed graph of n vertices with integer weights in $\{-M, \dots, 0, \dots, M\}$, where $M = O(1)$, is $O(n^{0.2096})$.*

Table 1: Summary of the applications of our algebraic techniques to graph-theoretic problems in the congested clique model. Here n both represents the number of vertices in the input graph and the number of nodes in the network.

Problem	Round complexity	Previously
APSP (undirected graphs, weights in $\{0, 1, \dots, M\}$)	$\tilde{O}\left(M^{\frac{2}{\omega}} n^{1-\frac{2}{\omega}}\right)$ Th. 4	$\tilde{O}\left(M n^{1-\frac{2}{\omega}}\right)$
APSP (directed graphs, constant weights)	$O(n^{0.2096})$ Th. 5	$\tilde{O}(n^{1/3})$
Diameter (undirected graphs, weights in $\{0, 1, \dots, M\}$)	$\tilde{O}\left(M^{\frac{2}{\omega}} n^{1-\frac{2}{\omega}}\right)$ Cor. 1	$\tilde{O}\left(M n^{1-\frac{2}{\omega}}\right)$
Computing the size of a maximum matching	$O(n^{1-\frac{2}{\omega}} \log n)$ Th. 8	—
Computing allowed edges in a perfect matching	$O(n^{1-1/\omega})$ Sec. 6.3	—
Gallai-Edmonds decomposition	$O(n^{1-1/\omega})$ Th. 9	—
Minimum vertex cover in bipartite graphs	$O(n^{1-1/\omega})$ Sec. 6.4	—

Prior to this work, the upper bound for the round complexity of this problem was $\tilde{O}(n^{1/3})$, obtained by directly computing the distance product (as done in [5]) in the congested clique model. Again, Theorem 5 follows easily from Theorem 7 and the observation that the reduction to distance product computation given in [47] can be implemented efficiently in the congested clique model. The exponent 0.2096 in the statement of Theorem 5 is derived from the current best upper bounds on the complexity of rectangular matrix multiplication in the centralized setting [24].

Theorems 2 and 3 also enable us to solve a multitude of graph-theoretic problems in the congested clique model with a sublinear number of rounds. Examples described in this paper are computing the number of edges in a maximum matching of a simple graph with $O(n^{1-2/\omega} \log n)$ rounds, computing the set of allowed edges in a perfect matching, the Gallai-Edmonds decomposition of a simple graph, and a minimum vertex cover in a bipartite graph with $O(n^{1-1/\omega})$ rounds. These results are obtained almost immediately from the appropriate reductions to matrix inversion and similar problems known the centralized setting [7, 31, 38] — indeed it is not hard to adapt all these reductions so that they can be implemented efficiently in the congested clique model. Note that while non-algebraic centralized algorithms solving these problems also exist (see, e.g., [32]), they are typically sequential and do not appear to be efficiently implementable in the congested clique model. The algebraic approach developed in this paper, made possible by our algorithms for the computation of the determinant, the rank and the inverse of matrix, appears to be currently the only way of obtaining fast algorithms for these problems in the congested clique model.

2 Preliminaries

Notations. Through this paper we will use n to denote the number of nodes in the network. The n nodes will be denoted $1, 2, \dots, n$. The symbol \mathbb{F} will always denote a finite field of order upper bounded by a polynomial in n (which means that each field element can be encoded with $O(\log n)$ bits and thus sent using one message in the congested clique model). Given any positive integer p , we use the notation $[p]$ to represent the set $\{1, 2, \dots, p\}$. Given any $p \times p'$ matrix A , we will write its entries as $A[i, j]$ for $(i, j) \in [p] \times [p']$, and use the notation $A[i, *]$ to represent its i -th row and $A[*, j]$ to represent its j -th column.

Graph-theoretic problems in the congested clique model. As mentioned in the introduction, typically the main tasks that we want to solve in the congested clique model are graph-theoretical problems. In all the applications given in this paper the number of vertices of the graph will be n , the same as the number of nodes of the network. The input will be given as follows: initially each node $\ell \in [n]$ has the ℓ -th row and the ℓ -th column of the adjacency matrix of the graph. Note that this distribution of the input, while being the most natural, is not essential; the only important assumption is that the entries are evenly distributed among the n nodes since they can then be redistributed in a constant number of rounds as shown in the following Lemma by Dolev et al. [12], which we will use many times in this paper.

Lemma 1. [12] *In the congested clique model a set of messages in which no node is the source of more than n messages and no node is the destination of more than n messages can be delivered within two rounds if the source and destination of each message is known in advance to all nodes.*

Algebraic problems in the congested clique model. The five main algebraic problems that we consider in this paper are defined as follows.

MM(n, m, k, \mathbb{F}) — Multiple Rectangular Matrix Multiplications

Input: matrices $A_1, \dots, A_k \in \mathbb{F}^{n \times m}$ and $B_1, \dots, B_k \in \mathbb{F}^{m \times n}$ distributed among the n nodes
(Node $\ell \in [n]$ has $A_1[\ell, *], \dots, A_k[\ell, *]$ and $B_1[*], \dots, B_k[*]$)
Output: the matrices $A_1 B_1, \dots, A_k B_k$ distributed among the n nodes
(Node $\ell \in [n]$ has $A_1 B_1[\ell, *], \dots, A_k B_k[\ell, *]$ and $A_1 B_1[*], \dots, A_k B_k[*]$)

DET(n, \mathbb{F}) — Determinant

Input: matrix $A \in \mathbb{F}^{n \times n}$ distributed among the n nodes (Node $\ell \in [n]$ has $A[\ell, *]$ and $A[*], \ell]$)
Output: $\det(A)$ (Each node of the network has $\det(A)$)

Rank(n, \mathbb{F}) — Rank

Input: matrix $A \in \mathbb{F}^{n \times n}$ distributed among the n nodes (Node $\ell \in [n]$ has $A[\ell, *]$ and $A[*], \ell]$)
Output: $\text{rank}(A)$ (Each node of the network has $\text{rank}(A)$)

INV(n, \mathbb{F}) — Inversion

Input: invertible matrix $A \in \mathbb{F}^{n \times n}$ distributed among the n nodes
(Node $\ell \in [n]$ has $A[\ell, *]$ and $A[*], \ell]$)
Output: matrix A^{-1} distributed among the n nodes (Node $\ell \in [n]$ has $A^{-1}[\ell, *]$ and $A^{-1}[*], \ell]$)
(Node $\ell \in [n]$ has $A^{-1}[\ell, *]$ and $A^{-1}[*], \ell]$)

SYS(n, \mathbb{F}) — Solution of a linear system

Input: invertible matrix $A \in \mathbb{F}^{n \times n}$ and vector $b \in \mathbb{F}^{n \times 1}$, distributed among the n nodes
(Node $\ell \in [n]$ has $A[\ell, *]$, $A[*], \ell]$ and b)
Output: the vector $x \in \mathbb{F}^{n \times 1}$ such that $Ax = b$ (Node $\ell \in [n]$ has $x[\ell]$)

Note that the distribution of the inputs and the outputs assumed in the above five problems is mostly chosen for convenience. For instance, if needed the whole vector x in the output of **SYS(n, \mathbb{F})** can be sent to all the nodes of the network in two rounds using Lemma 1. The only important assumption is that when dealing with matrices, the entries of the matrices must be evenly distributed among the n nodes.

We will also in this paper consider the distance product of two matrices, defined as follows.

Definition 1. Let m and n be two positive integers. Let A be an $n \times m$ matrix and B be an $m \times n$ matrix, both with entries in $\mathbb{R} \cup \{\infty\}$. The distance product of A and B , denoted $A * B$, is the $n \times n$ matrix C such that $C[i, j] = \min_{s \in [m]} \{A[i, s] + B[s, j]\}$ for all $(i, j) \in [n] \times [n]$.

We will be mainly interested in the case when the matrices have integer entries. More precisely, we will consider the following problem.

DIST(n, m, M) — Computation of the distance product

Input: an $n \times m$ matrix A and an $m \times n$ matrix B , with entries in $\{-M, \dots, -1, 0, 1, \dots, M\} \cup \{\infty\}$
(Node $\ell \in [n]$ has $A[\ell, *]$ and $B[*, \ell]$)

Output: the matrix $C = A * B$ distributed among the n nodes
(Node $\ell \in [n]$ has $C[\ell, *]$ and $C[*, \ell]$)

Centralized algebraic algorithms for matrix multiplication. We now briefly describe algebraic algorithms for matrix multiplication and known results about the complexity of rectangular matrix multiplication. We refer to [4] for a detailed exposition of these concepts.

Let \mathbb{F} be a field and m, n be two positive integer. Consider the problem of computing the product of an $n \times m$ matrix by an $m \times n$ matrix over \mathbb{F} . An algebraic algorithm for this problem is described by three sets $\{\alpha_{ij\mu}\}$, $\{\beta_{ij\mu}\}$ and $\{\lambda_{ij\mu}\}$ of coefficients from \mathbb{F} such that, for any $n \times m$ matrix A and any $m \times n$ matrix B , the equality

$$C[i, j] = \sum_{\mu=1}^t \lambda_{ij\mu} S^{(\mu)} T^{(\mu)}$$

holds for all $(i, j) \in [n] \times [n]$, where $C = AB$ and

$$S^{(\mu)} = \sum_{i=1}^n \sum_{j=1}^m \alpha_{ij\mu} A[i, j], \quad T^{(\mu)} = \sum_{i=1}^n \sum_{j=1}^m \beta_{ij\mu} B[j, i],$$

for each $s \in [t]$. Note that each $S^{(\mu)}$ and each $T^{(\mu)}$ is an element of \mathbb{F} . The integer t is called the rank of the algorithm, and corresponds to the complexity of the algorithm.

For instance, consider the trivial algorithm computing this matrix product using the formula

$$C[i, j] = \sum_{s=1}^m A[i, s] B[s, j].$$

This algorithm can be described in the above formalism by taking $t = n^2 m$, writing each $\mu \in [n^2 m]$ as a triple $\mu = (i', j', s') \in [n] \times [n] \times [m]$, and choosing

$$\lambda_{ij(i', j', s')} = \begin{cases} 1 & \text{if } i = i' \text{ and } j = j', \\ 0 & \text{otherwise,} \end{cases} \quad \alpha_{ij(i', j', s')} = \begin{cases} 1 & \text{if } i = i' \text{ and } j = s', \\ 0 & \text{otherwise,} \end{cases} \quad \beta_{ij(i', j', s')} = \begin{cases} 1 & \text{if } i = j' \text{ and } j = s', \\ 0 & \text{otherwise.} \end{cases}$$

Note that this trivial algorithm, and the description we just gave, also works over any semiring.

The exponent of matrix multiplication. For any non-negative real number γ , let $\omega(\gamma)$ denote the minimal value τ such that the product of an $n \times \lceil n^\gamma \rceil$ matrix over \mathbb{F} by an $\lceil n^\gamma \rceil \times n$ matrix over \mathbb{F} can be computed by an algebraic algorithm of rank $n^{\tau+o(1)}$ (i.e., can be computed with complexity $O(n^{\tau+\epsilon})$ for any $\epsilon > 0$). As usual in the literature, we typically abuse notation and simply write that such a product can be done with complexity $O(n^{\omega(\gamma)})$, i.e., ignoring the $o(1)$ in the exponent. The value $\omega(1)$ is denoted by ω , and often called the exponent of square matrix multiplication. Another important quantity is the value $\alpha = \sup\{\gamma \mid \omega(\gamma) = 2\}$.

The trivial algorithm for matrix multiplication gives the upper bound $\omega(\gamma) \leq 2 + \gamma$, and thus $\omega \leq 3$ and $\alpha \geq 0$. The current best upper bound on ω is $\omega < 2.3729$, see [25, 43]. The current best bound on α is $\alpha > 0.3029$, see [24]. The best bounds on $\omega(\gamma)$ for $\gamma > \alpha$ can also be found in [24].

3 Matrix Multiplication in the Congested Clique Model

In this section we present our results on the round complexity of $\text{MM}(n, m, k, \mathbb{F})$ and $\text{DIST}(n, m, M)$.

We first give the complete statement of our main result concerning $\text{MM}(n, m, k, \mathbb{F})$ that was stated in a simplified form in the introduction.

Theorem 1 (Complete version). *For any positive integer $k \leq n$, the deterministic round complexity of $\text{MM}(n, m, k, \mathbb{F})$ is*

$$\begin{cases} O(k) & \text{if } 0 \leq m \leq \sqrt{kn}, \\ O(k^{2/\omega(\gamma)} n^{1-2/\omega(\gamma)}) & \text{if } \sqrt{kn} \leq m < n^2/k, \\ O(km/n) & \text{if } m \geq n^2/k, \end{cases}$$

where γ is the solution of the equation

$$\left(1 - \frac{\log k}{\log n}\right) \gamma = 1 - \frac{\log k}{\log n} + \left(\frac{\log m}{\log n} - 1\right) \omega(\gamma). \quad (1)$$

For any $k \geq n$, the deterministic round complexity of $\text{MM}(n, m, k, \mathbb{F})$ is

$$\begin{cases} O(k) & \text{if } 1 \leq m \leq n, \\ O(km/n) & \text{if } m \geq n. \end{cases}$$

The proof of Theorem 1, which will also show that Equation (1) always has a solution when $k \leq n$ and $\sqrt{kn} \leq m < n^2/k$, is given in Section 3.1 (a short discussion of the proof ideas was presented in the introduction). The upper bounds we obtain for the case $k = 1$ are depicted in Figure 1, where Equation (1) is solved using the best known upper bound on $\omega(\gamma)$ from [24]. As briefly mentioned in the introduction, the round complexity is constant for any $k \leq \sqrt{n}$, and we further have round complexity $O(n^\epsilon)$, for any $\epsilon > 0$, for all values $k \leq n^{(1+\alpha)/2}$ (the bound $\alpha > 0.3029$ implies $(1 + \alpha)/2 > 0.6514$). For the case $m = n$ the solution of Equation (1) is $\gamma = 1$, which gives the bounds of the simplified version of Theorem 1 presented in the introduction.

We now give lower bounds on the round complexity of $\text{MM}(n, m, k, \mathbb{F})$ that show that the upper bounds of Theorem 1 are tight, except possibly in the case $\sqrt{kn} \leq m < n^2/k$ when $k \leq n$.

Proposition 1. *The randomized round complexity of $\text{MM}(n, m, k, \mathbb{F})$ is*

$$\begin{cases} \Omega(k) & \text{if } 1 \leq m \leq n, \\ \Omega(km/n) & \text{if } m \geq n. \end{cases}$$

Proof. We first prove the lower bound $\Omega(km/n)$ for any $m \geq n$. Let us consider instances of $\text{MM}(n, m, k, \mathbb{F})$ of the following form: for each $s \in [k]$ all the rows of A_s are zero except the first row; for each $s \in [k]$ all the columns of B_s are zero except the second column. Let us write $C_s = A_s B_s$ for each $s \in [k]$. We prove the lower bound by partitioning the n nodes of the network into the two sets $\{1\}$ and $\{2, \dots, n\}$, and considering the following two-party communication problem. Alice (corresponding to the set $\{1\}$) has for input $A_s[1, j]$ for all $j \in [m]$ and all $s \in [k]$. Bob (corresponding to the set $\{2, \dots, n\}$) has for input $B_s[i, 2]$ for all $i \in [m]$ and all $s \in [k]$. The goal is for Alice to output $C_s[1, 2]$ for all $s \in [k]$. Note that $C_s[1, 2]$ is the inner product (over \mathbb{F}) of the first row of A_s and the second column of B_s . Thus $\sum_{s=1}^k C_s[1, 2]$ is the inner product of two vectors of size km . Alice and Bob must exchange $\Omega(km \log |\mathbb{F}|)$ bits to compute this value [8], which requires $\Omega(km/n)$ rounds in the original congested clique model.

We now prove the lower bound $\Omega(k)$ for any $m \geq 1$. Let us consider instances of $\text{MM}(n, m, k, \mathbb{F})$ of the following form: for each $s \in [k]$, all entries of A_s are zero except the entry $A_s[1, 1]$ which is one; for each $s \in [k]$, $B_s[i, j] = 0$ for all $(i, j) \notin \{(1, j) \mid j \in \{2, \dots, n\}\}$ (the other $n - 1$ entries are arbitrary). Again, let us write $C_s = A_s B_s$ for each $s \in [k]$. We prove the lower bound by again partitioning the n nodes of the network into the two sets $\{1\}$ and $\{2, \dots, n\}$, and considering the following two-party communication problem. Alice has no input. Bob has for input $B_s[1, j]$ for all $j \in \{2, \dots, n\}$ and all $s \in [k]$. The goal is for Alice to output $C_s[1, j]$ for all $j \in \{2, \dots, n\}$ and all $s \in [k]$. Since the output reveals Bob's whole input to Alice, Alice must receive $\Omega(k(n - 1) \log |\mathbb{F}|)$ bits, which gives round complexity $\Omega(k)$ in the original congested clique model. \square

3.1 Proof of Theorem 1

Let us first prove the following proposition that deals with the case where m is large. In this case the algorithm is relatively simple.

Proposition 2. *For any $k \leq n$ and any $m \geq n^2/k$, the deterministic round complexity of $\text{MM}(n, m, k, \mathbb{F})$ is $O(km/n)$.*

Proof. We assume below for convenience that both n/k and km/n are integers. If this is not the case the proof can be adjusted in a straightforward manner by replacing them by $\lceil n/k \rceil$ and $\lceil km/n \rceil$, respectively.

For each $s \in [k]$, we will write $C_s = A_s B_s$. Let us decompose the matrix A_s into n/k matrices of size $n \times \frac{km}{n}$ by partitioning the m columns of A_s into n/k consecutive blocks of size km/n . Let us call these smaller matrices $A_s^{(1)}, \dots, A_s^{(n/k)}$. Similarly, for each $s \in [k]$ we decompose the matrix B_s into n/k matrices of size $\frac{km}{n} \times n$ by partitioning the m rows of B_s into n/k consecutive blocks of size km/n . Let us call these smaller matrices $B_s^{(1)}, \dots, B_s^{(n/k)}$. For each $t \in [n/k]$ we write $C_s^{(t)} = A_s^{(t)} B_s^{(t)}$.

For each $s \in [k]$, and each $i, j \in [n]$ the equation

$$C_s[i, j] = \sum_{t=1}^{n/k} C_s^{(t)}[i, j] \quad (2)$$

obviously holds. Our distributed algorithm is based on this simple observation. Each node of the network will be assigned, besides its original label $\ell \in [n]$, a second label $(s, t) \in [k] \times [n/k]$. The assignment of these labels is arbitrary, the only condition being that distinct nodes are assigned distinct labels. The distributed algorithm is as follows.

1. Node $(s, t) \in [k] \times [n/k]$ receives the whole two matrices $A_s^{(t)}$ and $B_s^{(t)}$ from the nodes of the networks owning the entries of these two matrices, and then locally computes $C_s^{(t)}$.
2. Node $\ell \in [n]$ receives $C_s^{(t)}[\ell, *]$ and $C_s^{(t)}[* , \ell]$ for all $s \in [k]$ and all $t \in [n/k]$ from the nodes of the network owning these entries, and then locally computes $C_s[\ell, *]$ and $C_s[* , \ell]$ using Equation (2) for all $s \in [k]$.

The number of field elements received per node is $2km$ at Step 1 and $2n^2$ at Step 2. The total number of field elements received per node is thus $O(km + n^2)$, which gives round complexity $O(km/n)$ when $km \geq n^2$ from Lemma 1. \square

The main technical contribution is the following proposition.

Proposition 3. *For any k and m such that $k \leq n$ and $\sqrt{kn} \leq m < n^2/k$, the deterministic round complexity of $\text{MM}(n, m, k, \mathbb{F})$ is*

$$O(k^{2/\omega(\gamma)} n^{1-2/\omega(\gamma)}),$$

where γ is the solution of the equation $(1 - \frac{\log k}{\log n})\gamma = 1 - \frac{\log k}{\log n} + (\frac{\log m}{\log n} - 1)\omega(\gamma)$.

Proof. For convenience, let us assume that n/k is an integer (otherwise we replace this value by the nearest integer). Let $\gamma \geq 0$ be a value that will be set later. Let d be the largest integer such that the product of a $d \times \lceil d^\gamma \rceil$ matrix by a $\lceil d^\gamma \rceil \times d$ matrix can be computed by an algebraic algorithm (as in Section 2) of rank n/k . Note that

$$d = \Theta\left((n/k)^{1/\omega(\gamma)}\right)$$

from the definition of the exponent of matrix multiplication. For convenience, we assume below that n/d , $\sqrt{n/k}$ and \sqrt{kn}/d are also integers (otherwise we can again simply replace these values by the nearest integer). Define the quantity

$$r = \frac{m}{d^\gamma \sqrt{n/k}} = \Theta\left(\frac{m}{(n/k)^{\gamma/\omega(\gamma)+1/2}}\right).$$

Our choice of γ (discussed later) will guarantee that $m \geq (n/k)^{\gamma/\omega(\gamma)+1/2}$, which implies $r = \Omega(1)$. We will thus assume below, for convenience but without affecting the analysis of the complexity of the algorithm, that r and d^γ are integers such that $r \geq 1$ and $d^\gamma \leq m$, and that m/d^γ is an integer as well.

For each $s \in [k]$, we write the entries of A_s and B_s as $A_s[ix, jz]$ and $B_s[jz, ix]$, respectively, where $i \in [d]$, $j \in [d^\gamma]$, $x \in [n/d]$ and $z \in [m/d^\gamma]$. More precisely, this notation corresponds to decomposing each row of A_s and each column of B_s into d consecutive blocks of n/d entries, and decomposing each column of A_s and each row of B_s into d^γ consecutive blocks of m/d^γ entries. Note that this corresponds to decomposing A_s into an $d \times d^\gamma$ matrix where each entry is a submatrix of A_s of size $(n/d) \times (m/d^\gamma)$, and decomposing B_s into an $d^\gamma \times d$ matrix where each entry is a submatrix of B_s of size $(m/d^\gamma) \times (n/d)$. We will write $A_s[i*, j*]$ and $B_s[j*, i*]$ to represent these submatrices. Similarly, we write the elements of the output matrix $C_s = A_s B_s$ as $C_s[ix, jy]$, where $i, j \in [d]$ and $x, y \in [n/d]$. By extending the formulation given in Section 2 to block matrices (and using the same notations), for any $s \in [k]$, any $i, j \in [d]$ and any $x, y \in [n/d]$ we can write

$$C_s[ix, jy] = \sum_{\mu=1}^{n/k} \lambda_{ij\mu} S_s^{(\mu)} T_s^{(\mu)}, \quad (3)$$

where, for each $\mu \in [n]$, $S_s^{(\mu)}$ is an $(n/d) \times (m/d^\gamma)$ matrix that can be written as a linear combination of the submatrices of A_s , and $T_s^{(\mu)}$ is an $(m/d^\gamma) \times (n/d)$ matrix that can be written as a linear combination of the submatrices of B_s :

$$S_s^{(\mu)} = \sum_{i=1}^d \sum_{j=1}^{d^\gamma} \alpha_{ij\mu} A_s[i*, j*] \quad (4)$$

$$T_s^{(\mu)} = \sum_{i=1}^d \sum_{j=1}^{d^\gamma} \beta_{ij\mu} B_s[j*, i*]. \quad (5)$$

Finally, we will also decompose each label x , y and z into two parts:

- we write $x = wx'$ where $w \in [\sqrt{n/k}]$ and $x' \in [\sqrt{kn}/d]$,
- we write $y = wy'$ where $w \in [\sqrt{n/k}]$ and $y' \in [\sqrt{kn}/d]$,
- we write $z = wz'$ where $w \in [\sqrt{n/k}]$ and $z' \in [r]$.

This corresponds to further decomposing each submatrix of A_s into an $\sqrt{n/k} \times \sqrt{n/k}$ matrix where each entry is a smaller submatrix of A_s of size $(\sqrt{kn}/d) \times r$, and decomposing each submatrix of B_s into an $\sqrt{n/k} \times \sqrt{n/k}$ matrix where each entry is a smaller submatrix of B_s of size $r \times (\sqrt{kn}/d)$.

Each node of the network has a label $(i, x) \in [d] \times [n/d]$, and receives as input $A_s[ix, *]$ and $B_s[*, ix]$ for all $s \in [k]$. We assign two additional labels to each node: one label $(s, u, v) \in [k] \times [\sqrt{n/k}] \times [\sqrt{n/k}]$ and one label $(s, \mu) \in [k] \times [n/k]$. The assignment of these labels is arbitrary, the only condition being that distinct nodes are assigned distinct labels. The algorithm is given in Figure 2.

The number of field elements received per node at Step 1 is

$$2 \times d \times d^\gamma \times \frac{\sqrt{kn}}{d} \times \frac{m}{d^\gamma \sqrt{n/k}} = 2km.$$

The number of field elements received per node at Step 2 is

$$2 \times \frac{n}{d} \times \frac{m}{d^\gamma} = O\left(k^{(1+\gamma)/\omega(\gamma)} mn^{1-(1+\gamma)/\omega(\gamma)}\right). \quad (6)$$

The number of field elements received per node at Step 3 is

$$\frac{n}{k} \times \frac{\sqrt{kn}}{d} \times \frac{\sqrt{kn}}{d} = O\left(k^{2/\omega(\gamma)} n^{2-2/\omega(\gamma)}\right). \quad (7)$$

The number of field elements received per node at Step 4 is $2kn$. Note that Expression (6) is larger than $2km$ since $\omega(\gamma) \geq 1+\gamma$ and $k \leq n$. Moreover, Expression (7) is larger than $2kn$ since $\omega(\gamma) \geq 2$ and $k \leq n$. Thus the total number of field elements received per node is

$$O\left(k^{(1+\gamma)/\omega(\gamma)} mn^{1-(1+\gamma)/\omega(\gamma)} + k^{2/\omega(\gamma)} n^{2-2/\omega(\gamma)}\right).$$

Let us write $a = \log k / \log n$ and $b = \log m / \log n$ and define the two functions

$$f(\gamma) = \frac{a(1+\gamma)}{\omega(\gamma)} + b + 1 - \frac{1+\gamma}{\omega(\gamma)},$$

$$g(\gamma) = \frac{2a}{\omega(\gamma)} + 2 - \frac{2}{\omega(\gamma)},$$

1. Node $(s, u, v) \in [k] \times [\sqrt{n/k}] \times [\sqrt{n/k}]$ receives $A_s[iux', jvz']$ and $B_s[juz', ivy']$ from the nodes of the network owning these entries, for all $i \in [d]$, $j \in [d]$, $x', y' \in [\sqrt{kn}/d]$ and $z' \in [r]$. Node (s, u, v) then locally computes $S_s^{(\mu)}[ux', vz']$ using Equation (4) and $T_s^{(\mu)}[uz', vy']$ using Equation (5) for all $\mu \in [n/k]$, all $x', y' \in [\sqrt{kn}/d]$ and all $z' \in [r]$.
2. Node $(s, \mu) \in [k] \times [n/k]$ receives the whole matrices $S_s^{(\mu)}$ and $T_s^{(\mu)}$ from the nodes of the network owning the entries of these matrices, and locally computes the product $P_s^{(\mu)} = S_s^{(\mu)} T_s^{(\mu)}$.
3. Node $(s, u, v) \in [k] \times [\sqrt{n/k}] \times [\sqrt{n/k}]$ receives $P_s^{(\mu)}[ux', vy']$ from the nodes of the network owning these entries, for all $\mu \in [n/k]$ and all $x', y' \in [\sqrt{kn}/d]$. Node (s, u, v) then locally computes $C_s[iux', jvy']$ using Equation (3) for all $i, j \in [d]$ and all $x', y' \in [\sqrt{kn}/d]$.
4. Node $(i, x) \in [d] \times [n/d]$ receives $C_s[ix, *]$ and $C_s[*, ix]$ for all $s \in [k]$ from the nodes of the network owning these entries.

Figure 2: Distributed algorithm for $\text{MM}(n, m, k, \mathbb{F})$ in the congested clique model. Initially each node $(i, x) \in [d] \times [n/d]$ has as input $A_s[ix, *]$ and $B_s[*, ix]$ for all $s \in [k]$.

for any $\gamma \geq 0$. The function f is a decreasing function of γ , with value $a/2 + b + 1/2$ when $\gamma = 0$ and with limit $a + b$ when γ goes to infinity. The function g is an increasing function of γ , with value $a + 1$ when $\gamma = 0$ and with limit 2 when γ goes to infinity. Let us choose γ as follows. Since we assumed $\sqrt{kn} \leq m < n^2/k$, the equation $f(\gamma) = g(\gamma)$ necessarily has a solution. We choose γ as this solution, i.e., such that

$$(1 - a)\gamma = 1 - a + (b - 1)\omega(\gamma). \quad (8)$$

Observe that such a choice for γ implies that $m \geq (n/k)^{\gamma/\omega(\gamma)+1/2}$ as we required: Equation (8) can be rewritten as

$$b = 1 + \frac{(1 - a)\gamma}{\omega(\gamma)} - \frac{1 - a}{\omega(\gamma)},$$

which implies

$$b \geq (1 - a) \left(\frac{\gamma}{\omega(\gamma)} + \frac{1}{2} \right)$$

since $\omega(\gamma) \geq 2$. For our choice of γ , the total number of field elements received per node is thus $O(k^{2/\omega(\gamma)} n^{2-2/\omega(\gamma)})$, which gives round complexity

$$O(k^{2/\omega(\gamma)} n^{1-2/\omega(\gamma)}),$$

as claimed, from Lemma 1. □

The proof of Theorem 1 now follows easily from Propositions 2 and 3.

Proof of Theorem 1. Consider first the case $k \leq n$. When $\sqrt{kn} \leq m \leq n^2/k$, we use the algorithm of Proposition 3. When $m \geq n^2/k$, we use the algorithm of Proposition 2. When $0 \leq m \leq \sqrt{kn}$ the

claimed upper bound $O(k)$ on the round complexity can be derived from Proposition 2 by taking $m = \sqrt{kn}$ (i.e., by appending rows and columns with zero entries to the input matrices) and $\gamma = 0$.

For the case $k \geq n$ we can simply repeat $\lceil k/n \rceil$ times an algorithm for $\text{MM}(n, m, n, \mathbb{F})$, which gives round complexity

$$\begin{cases} O(k) & \text{if } 1 \leq m \leq n, \\ O(km/n) & \text{if } m \geq n. \end{cases}$$

from the analysis of the previous paragraph. \square

3.2 Application to the distance product

One of the main applications of Theorem 1 is the following result, which will be the key ingredient for all our results on the all-pairs shortest paths and diameter computation discussed in Section 6.

Theorem 7. *For any $M \leq n$ and $m \leq n$, the deterministic round complexity of $\text{DIST}(n, m, M)$ is*

$$\begin{cases} O(M \log m) & \text{if } 0 \leq m \leq \sqrt{Mn \log m}, \\ O((M \log m)^{2/\omega(\gamma)} n^{1-2/\omega(\gamma)}) & \text{if } \sqrt{Mn \log m} \leq m \leq n^2/(M \log m), \\ O(mM \log m/n) & \text{if } n^2/(M \log m) \leq m \leq n, \end{cases}$$

where γ is the solution of the equation $(1 - \frac{\log M}{\log n}) \gamma = 1 - \frac{\log M}{\log n} + (\frac{\log m}{\log n} - 1) \omega(\gamma)$.

Let us first give a brief overview of the proof of Theorem 7. The idea is to show that $\text{DIST}(n, m, M)$ reduces to $\text{MM}(n, m, k, \mathbb{F})$ for $k \approx M \log m$ and a well-chosen finite field \mathbb{F} , and then use Theorem 1 to get a factor $(M \log m)^{2/\omega(\gamma)}$, instead of the factor M obtained in a straightforward implementation of the distance product, in the complexity. This reduction is done by first applying a standard encoding of the distance product into a usual matrix product of matrices with integer entries of absolute value $\exp(M)$, and then using Fourier transforms to split this latter matrix product into roughly $M \log m$ independent matrix products over a small field.

Proof of Theorem 7. We show below that $\text{DIST}(n, m, M)$ reduces to $\text{MM}(n, m, k, \mathbb{F})$ where $k = O(M \log m)$ and $|\mathbb{F}| = \text{poly}(M, \log m)$. The result then follows from Theorem 1.

Let N be any positive integer. We first show how to reduce the multiplication or the addition of two nonnegative N -bit integers a and b to $2N$ independent operations (multiplications or additions, respectively) in a large enough field, using the Fourier transform. Let \mathbb{F} be a finite field with characteristic at least $N + 1$. Let

$$\Phi_N: \{0, 1, \dots, 2^{N-1}\} \rightarrow \mathbb{F}[x]/(x^{2N} - 1)$$

be the map that maps each N -bit integer $c = \sum_{i=0}^{N-1} c_i 2^i$, with each c_i in $\{0, 1\}$, to the polynomial $\sum_{i=0}^{N-1} c_i x^i$. Computing the product $ab \in \mathbb{Z}$ (resp. the sum $a + b \in \mathbb{Z}$) reduces to computing the product $\Phi_N(a)\Phi_N(b)$ (resp. the sum $\Phi_N(a) + \Phi_N(b)$) over $\mathbb{F}[x]/(x^{2N} - 1)$. Indeed, if $\Phi_N(a)\Phi_N(b)$ is the polynomial $\sum_{i=0}^{2N-1} c_i x^i$, with each c_i in \mathbb{F} , then ab is equal to the sum $\sum_{i=0}^{2N-1} c_i 2^i$ computed over the integers (note that the assumption on the characteristic of \mathbb{F} is crucial here), and similarly for the addition. The computation of $\Phi_N(a)\Phi_N(b)$ and $\Phi_N(a) + \Phi_N(b)$ can be done using the identities

$$\Phi_N(a)\Phi_N(b) = \text{DFT}^{-1}(\text{DFT}(\Phi_N(a)) \cdot \text{DFT}(\Phi_N(b))), \quad (9)$$

$$\Phi_N(a) + \Phi_N(b) = \text{DFT}^{-1}(\text{DFT}(\Phi_N(a)) + \text{DFT}(\Phi_N(b))), \quad (10)$$

where $DFT: \mathbb{F}[x]/(x^{2N} - 1) \rightarrow \mathbb{F}^{2N}$ is the Fourier transform and where \cdot and the second $+$ represent the coordinate-wise multiplication and addition in \mathbb{F}^{2N} , respectively (we refer to [4] for a detailed presentation of this Fourier transform). In order for the Fourier transform to be defined we nevertheless need to choose the field \mathbb{F} such that it contains a $2N$ -th primitive root of unity. It is known (see, e.g., [11]) that for any prime p such that $2N$ divides $p - 1$ the finite field $\mathbb{F} = \mathbb{Z}_p$ contains such a prime root. It is also known ([28, 29], see also [11]) that the least prime p such that $2N$ divides $p - 1$ is smaller than $d(2N)^{d'}$ for some absolute constants d and d' . By choosing such a prime p , we obtain a reduction from the computation of ab (resp. $a + b$) to one coordinate-wise multiplication (resp. addition) in \mathbb{F}^{2N} where $|\mathbb{F}| = \text{poly}(N)$. Note that we do not need to discuss the costs of finding p and the cost of preprocessing/postprocessing operations (such as applying the Fourier transform and its inverse), since they will have no impact on the round complexity of the algorithm we design below.

Let us now consider the task of computing the integer $\sum_{t=1}^m a_t b_t$ given non-negative N -bit integers $a_1, \dots, a_m, b_1, \dots, b_m$. Similarly to what we just did, this sum can be recovered from the polynomial $\sum_{t=1}^m \Phi_N(a_t) \Phi_N(b_t)$, which can be obtained by computing the $2N$ -dimensional vector

$$\sum_{t=1}^m DFT(\Phi_N(a_t)) \cdot DFT(\Phi_N(b_t)) \quad (11)$$

over a finite field \mathbb{F} of order polynomial in N and m , and then applying the inverse Fourier transform.

We can now describe our reduction. Let A, B be the matrices with entries in $\{-M, \dots, M\} \cup \{\infty\}$ of which we want to compute the distance product. We first reduce this distance product to one usual product of matrices with large entries, using standard techniques [2, 42, 47]. Consider the $n \times m$ matrix A' by the $m \times n$ matrix B' defined as:

$$A'[i, j] = \begin{cases} (m+1)^{M-A[i, j]} & \text{if } A[i, j] \neq \infty, \\ 0 & \text{if } A[i, j] = \infty, \end{cases} \quad B'[j, i] = \begin{cases} (m+1)^{M-B[j, i]} & \text{if } B[j, i] \neq \infty, \\ 0 & \text{if } B[j, i] = \infty, \end{cases}$$

for all $(i, j) \in [n] \times [m]$. It is easy to check (see [47] for a proof) that the entry $A * B[i, j]$ can be recovered easily from entry $A'B'[i, j]$, for each $(i, j) \in [n] \times [m]$. Let $N = \lceil \log_2((m+1)^{2M} + 1) \rceil$ be the number of bits needed to represent the entries of A' and B' , and \mathbb{F} be a finite field as in the previous paragraph. Next, in order to compute $A'B'$ we use the following strategy. For each $(i, j) \in [n] \times [m]$, consider the two vectors $DFT(\Phi_N(A'[i, j])) \in \mathbb{F}^{2N}$ and $DFT(\Phi_N(B'[j, i])) \in \mathbb{F}^{2N}$. For convenience write them as $\vec{p}_{ij} = (p_{ij}[1], \dots, p_{ij}[2N])$ and $\vec{q}_{ji} = (q_{ji}[1], \dots, q_{ji}[2N])$, respectively. Now, for any $s \in [2N]$, define the matrix $A'_s \in \mathbb{F}^{n \times m}$ and the matrix $B'_s \in \mathbb{F}^{m \times n}$ such that $A'_s[i, j] = p_{ij}[s]$ and $B'_s[j, i] = q_{ji}[s]$ for all $(i, j) \in [n] \times [m]$. It follows from the discussion of the previous paragraph (and in particular Equation (11)) that for each $(i, j) \in [n] \times [n]$ the entry $A'B'[i, j]$ can be recovered from the entries $A'_1 B'_1[i, j], \dots, A'_{2N} B'_{2N}[i, j]$. Since all preprocessing and postprocessing steps of this strategy can be performed locally by the nodes of the network in the congested clique model, this reduces the computation of $A'B'$ to solving one instance of $\text{MM}(n, m, 2N, \mathbb{F})$, with $2N = O(M \log m)$ and $|\mathbb{F}| = \text{poly}(M, m)$, as claimed. \square

4 Deterministic Computation of Determinant and Inverse Matrix

In this section we present deterministic algorithms for computing the determinant of a matrix and the inverse of a matrix in the congested clique model, and prove Theorem 2. Our algorithms can

be seen as efficient implementations of the parallel algorithm by Prerarata and Sarwate [37] based on the Faddeev-Leverrier method.

Let A be an $n \times n$ matrix over a field \mathbb{F} . Let $\det(\lambda I - A) = \lambda^n + c_1 \lambda^{n-1} + \dots + c_{n-1} \lambda + c_n$ be its characteristic polynomial. The determinant of A is $(-1)^n c_n$ and, if $c_n \neq 0$, its inverse is

$$A^{-1} = -\frac{A^{n-1} + c_1 A^{n-2} + \dots + c_{n-2} A + c_{n-1} I}{c_n}.$$

Define the vector $\vec{c} = (c_1, \dots, c_n)^T \in \mathbb{F}^{n \times 1}$. For any $k \in [n]$ let s_k denote the trace of the matrix A^k , and define the vector $\vec{s} = (s_1, \dots, s_n)^T \in \mathbb{F}^{n \times 1}$. Define the $n \times n$ matrix

$$S = \begin{pmatrix} 1 & & & & \\ s_1 & 2 & & & \\ s_2 & s_1 & 3 & & \\ \vdots & \vdots & \vdots & \ddots & \\ s_{n-1} & s_{n-2} & s_{n-3} & \dots & s_1 & n \end{pmatrix}.$$

It can be easily shown (see, e.g., [9, 37]) that $S\vec{c} = -\vec{s}$, which enables us to recover \vec{c} from \vec{s} if S is invertible. The matrix S is invertible whenever $n! \neq 0$, which is true in any field of characteristic zero or in any finite field of characteristic strictly larger than n . The following proposition shows that the inverse of an invertible triangular matrix can be computed efficiently in the congested clique model.

Proposition 4. *Let \mathbb{F} be any field. The deterministic round complexity of $\text{INV}(n, \mathbb{F})$, when the input A is an invertible lower triangular matrix, is $O(n^{1-2/\omega})$.*

Proof. We adapt the standard sequential algorithm for triangular matrix inversion [1, 3]. Let A be an invertible lower triangular $n \times n$ matrix with entries in \mathbb{F} . Assume without loss of generality that n is a power of two. Let us decompose A in four blocks of size $n/2 \times n/2$:

$$A = \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{pmatrix}.$$

Observe that both A_{11} and A_{22} are invertible lower triangular matrices, and

$$A^{-1} = \begin{pmatrix} A_{11}^{-1} & 0 \\ -A_{22}^{-1} A_{21} A_{11}^{-1} & A_{22}^{-1} \end{pmatrix}.$$

Inverting A thus reduces to inverting two invertible lower triangular matrices of size $n/2 \times n/2$ and performing two matrix multiplications. We implement this algorithm recursively (and in parallel) in the congested clique model as follows. The n nodes are partitioned into two groups of size $n/2$: the first group consisting of nodes $1, \dots, n/2$ and the second group consisting of nodes $n/2+1, \dots, n$. The first group recursively computes A_{11}^{-1} and the second group recursively computes A_{22}^{-1} . The important point here is that the computation of A_{11}^{-1} and the computation of A_{22}^{-1} can be done independently (i.e., in parallel). The nodes of the second group then in two rounds (using Lemma 1) distribute appropriately the rows of A_{22}^{-1} to the nodes of the first group (who own the columns of A_{21}), so that the nodes of the first group can compute $A_{22}^{-1} A_{21}$ and then $-A_{22}^{-1} A_{21} A_{11}^{-1}$. The nodes of the first group finally distribute appropriately the rows of $-A_{22}^{-1} A_{21} A_{11}^{-1}$ to the nodes of the second group, in two rounds from Lemma 1.

Let $R_I(n)$ denote the round complexity of our problem (computing the inverse of an invertible lower triangular $n \times n$ matrix using n nodes), and $R_M(n)$ denote the round complexity of computing the product of two $n \times n$ matrices using n nodes (i.e., the problem $\text{MM}(n, n, 1, \mathbb{F})$). The recurrence relation we obtain is

$$\begin{cases} R_I(1) = 0, \\ R_I(n) \leq R_I(n/2) + 2R_M(n/2) + 4 & \text{for } n \geq 2, \end{cases}$$

which gives $R_I(n) = O(n^{1-2/\omega})$ since $R_M(n) = O(n^{1-2/\omega})$ from Theorem 1. \square

We are now ready to give the proof of Theorem 2.

Proof of Theorem 2. For convenience we assume that n is a square, and write $p = \sqrt{n}$. If n is not a square we can easily adapt the proof by taking $p = \lceil \sqrt{n} \rceil$. Observe that any integer $a \in \{0, 1, \dots, n-1\}$ can be written in a unique way as $a = (a_1 - 1)p + (a_2 - 1)$ with $a_1, a_2 \in [p]$. Below when we write $a = (a_1, a_2) \in [n]$, we mean that a_1 and a_2 are the two elements in $[p]$ such that $a = (a_1 - 1)p + (a_2 - 1)$.

For any $\ell \in [n]$, let R_ℓ be the $p \times n$ matrix such that the i -th row of R_ℓ is the ℓ -th row of $A^{(i-1)p}$, for each $i \in [p]$. Similarly, for any $\ell \in [n]$, let C_ℓ be the $n \times p$ matrix such that the j -th column of C_ℓ is the ℓ -th column of A^{j-1} , for each $j \in [p]$. For each $\ell \in [n]$ define $U_\ell = R_\ell C_\ell$, which is a $p \times p$ matrix. Observe that, for any $k = (k_1, k_2) \in [n]$, the identity

$$s_k = \sum_{\ell=1}^n U_\ell[k_1, k_2] \quad (12)$$

holds. We will use this expression, together with the equation $\vec{c} = -S^{-1}\vec{s}$ to compute the determinant in the congested clique model.

In order to compute the inverse of A we then use the following approach. For any $(a_1, a_2) \in [p] \times [p]$, define the coefficient $c_{a_1, a_2} \in \mathbb{F}$ as follows:

$$c_{a_1, a_2} = \begin{cases} c_{n-1-(a_1-1)p-(a_2-1)} & \text{if } (a_1, a_2) \neq (p, p), \\ 1 & \text{if } (a_1, a_2) = (p, p). \end{cases}$$

For any $a_2 \in [p]$, define the $n \times n$ matrix E_{a_2} as follows:

$$E_{a_2} = \sum_{a_1=1}^p c_{a_1, a_2} A^{(a_1-1)p}.$$

Note that the following holds whenever $c_n \neq 0$:

$$A^{-1} = -\frac{\sum_{a=0}^{n-1} c_{n-1-a} A^a}{c_n} = -\frac{\sum_{a_1=1}^p \sum_{a_2=1}^p c_{a_2, a_1} A^{(a_1-1)p+(a_2-1)}}{c_n} = -\frac{\sum_{a_2=1}^p E_{a_2} A^{a_2-1}}{c_n}. \quad (13)$$

The algorithm for $\text{DET}(n, \mathbb{F})$ and $\text{INV}(n, \mathbb{F})$ is described in Figure 3. Steps 1 and 7.2 can be implemented in $O(p^{2/\omega} n^{1-2/\omega})$ rounds from Theorem 1 (or its simplified version in the introduction). Step 5 can be implemented in $O(n^{1-2/\omega})$ rounds, again from Theorem 1. At Steps 2, 3 and 6 each node receives n elements from the field \mathbb{F} , so each of these three steps can be implemented in two rounds from Lemma 1. The other steps (Steps 4, 7.1 and 7.3) do not require any communication. The total round complexity of the algorithm is thus $O(p^{2/\omega} n^{1-2/\omega}) = O(n^{1-1/\omega})$, as claimed. \square

1. The matrices $A^{(a_1-1)p}$ and A^{a_2-1} are computed for all $a_1, a_2 \in [p]$ using the distributed algorithm of Theorem 1. At the end of this step node $\ell \in [n]$ has the whole $p \times n$ matrix R_ℓ and the whole $n \times p$ matrix C_ℓ .
2. Node $\ell \in [n]$ locally computes U_ℓ , and sends $U_\ell[k_1, k_2]$ to each node $k = (k_1, k_2) \in [n]$.
3. Node $k = (k_1, k_2) \in [n]$, who received $U_\ell[k_1, k_2]$ for all $\ell \in [n]$ at the previous step, locally computes s_k using Equation (12). Node k then sends s_k to all the nodes.
4. Node $\ell \in [n]$, who received \vec{s} at Step 3, locally constructs $S[\ell, *]$ and $S[*, \ell]$.
5. The matrix S^{-1} is computed using the algorithm of Proposition 4. At the end of this step, node $\ell \in [n]$ has $S^{-1}[\ell, *]$ and $S^{-1}[*, \ell]$.
6. Node $\ell \in [n]$ locally computes c_ℓ from $S^{-1}[\ell, *]$ and \vec{s} , and sends c_ℓ to all nodes.
7. The determinant of A is $(-1)^n c_n$. If $c_n = 0$ the matrix A is not invertible. Otherwise the nodes compute A^{-1} as follows:
 - 7.1 Node $\ell \in [n]$ computes $E_{a_2}[\ell, *]$ for each $a_2 \in [p]$ (this can be done locally since \vec{c} and each row $A^{(a_1-1)p}[\ell, *]$ are known from Steps 6 and 1, respectively).
 - 7.2 The matrices $E_{a_2}A^{a_2-1}$ are computed for all $a_2 \in [p]$ using the distributed algorithm of Theorem 1 (since, besides $E_{a_2}[\ell, *]$ obtained at the previous step, each node $\ell \in [n]$ knows $A^{a_2-1}[*, \ell]$ from the result of the computation of Step 1). At the end of this step, node $\ell \in [n]$ has the ℓ -th row and the ℓ -th column of the matrix $E_{a_2}A^{a_2-1}$ for all $a_2 \in [p]$.
 - 7.3 Node $\ell \in [n]$ computes locally $A^{-1}[\ell, *]$ and $A^{-1}[*, \ell]$ using Equation (13).

Figure 3: Distributed algorithm for computing the determinant of an $n \times n$ matrix A and computing A^{-1} if $\det(A) \neq 0$. Initially each node $\ell \in [n]$ has as input $A[\ell, *]$ and $A[*, \ell]$.

5 Randomized Algorithms for Algebraic Problems

In this section we present $O(n^{1-2/\omega})$ -round randomized algorithms for computing the determinant, the rank and for solving linear systems of equations in the congested clique model, and prove Theorem 3. Our approach is based on Wiedemann's method [44] and its parallel implementations [21, 22, 23].

Let A be an $n \times n$ matrix over \mathbb{F} . The minimal polynomial of A , which we denote $\mathbf{minpol}(A)$, is the monic polynomial g over \mathbb{F} of least degree such that $g(A) = 0$. Let $v \in \mathbb{F}^{n \times 1}$ and $w \in \mathbb{F}^{1 \times n}$ be any vectors. Consider the sequence $wA^0v, wAv, wA^2v, \dots, wA^{n-1}v$ consisting of n elements of \mathbb{F} . This sequence is linearly generated over \mathbb{F} , and thus also admits a polynomial called the generating polynomial of the sequence, which we denote $\mathbf{minpol}(A, v, w)$. We refer to [16] for the precise definition of the generating polynomial of such a linearly generated sequence and a description of efficient algorithms to compute it — in this paper we will just need to know that such a polynomial exists. Wiedemann [44] showed that with high probability $\mathbf{minpol}(A, v, w) = \mathbf{minpol}(A)$ when v and w are chosen at random. We use the following characterization of this property proved by Kaltofen and Pan [22].

Lemma 2. ([22]) Let A be any $n \times n$ matrix over \mathbb{F} . Let $v \in \mathbb{F}^{n \times 1}$ and $w \in \mathbb{F}^{1 \times n}$ be two vectors in which each coordinate is chosen uniformly at random from \mathbb{F} . Then

$$\Pr [\mathbf{minpol}(A) = \mathbf{minpol}(A, v, w)] \geq 1 - \frac{2n}{|\mathbb{F}|}.$$

In general the minimal polynomial of a matrix A is not equal to its characteristic polynomial $\mathbf{charpol}(A)$. Wiedemann [44] nevertheless showed that the characteristic polynomial can be obtained from the minimal polynomial by preconditioning the matrix. Since it will be more convenient for our purpose to apply a diagonal preconditioner, we will use the following version shown by Chen et al. [6].

Lemma 3. ([6]) Let A be any $n \times n$ matrix over \mathbb{F} . Let D be an $n \times n$ diagonal matrix where each diagonal entry is chosen uniformly at random from $\mathbb{F} \setminus \{0\}$. Then

$$\Pr [\mathbf{charpol}(DA) = \mathbf{minpol}(DA)] \geq 1 - \frac{n(n-1)}{2(|\mathbb{F}| - 1)}.$$

Kaltofen and Saunders [23] showed that the rank of a matrix can be obtained, with high probability, from the degree of the minimal polynomial by using another preconditioning of the matrix. The following lemma follows from the combination of Theorem 1 and Lemma 2 in [23].

Lemma 4. ([23]) Let A be any $n \times n$ matrix over \mathbb{F} such that $\text{rank}(A) < n$. Let

$$U = \begin{pmatrix} 1 & u_2 & u_3 & \cdots & u_n \\ & 1 & u_2 & \cdots & u_{n-1} \\ & & 1 & \ddots & \vdots \\ & & & \ddots & u_2 \\ & & & & 1 \end{pmatrix}, \quad V = \begin{pmatrix} 1 & & & & \\ v_2 & 1 & & & \\ v_3 & v_2 & 1 & & \\ \vdots & & \ddots & \ddots & \\ v_n & v_{n-1} & \cdots & v_2 & 1 \end{pmatrix}$$

be two $n \times n$ unit (upper and lower, respectively) triangular random Toeplitz matrices. Let D be an $n \times n$ diagonal matrix where each diagonal entry is chosen uniformly at random from \mathbb{F} . Then

$$\Pr [\text{rank}(A) = \deg(\mathbf{minpol}(UAVD)) - 1] \geq 1 - \frac{n(3n+1)}{2|\mathbb{F}|}.$$

We are now ready to prove Theorem 3.

Proof of Theorem 3. We first show how to compute efficiently, in the congested clique model, the sequence $\tilde{A}^0 u, \tilde{A} u, \tilde{A}^2 u, \dots, \tilde{A}^{n-1} u$ given an arbitrary matrix $\tilde{A} \in \mathbb{F}^{n \times n}$ and an arbitrary vector $u \in \mathbb{F}^{n \times 1}$. For convenience assume that n is a power of two (otherwise we can simply add zero rows and columns to \tilde{A} and zero entries to u), and write $n = 2^k$ for some positive integer k . For each $i \in \{0, \dots, k\}$, define the $n \times n$ matrix

$$M^{(i)} = [u | \tilde{A} u | \cdots | \tilde{A}^{2^i - 1} u | 0 | \cdots | 0],$$

obtained by concatenating the vectors $u, \dots, \tilde{A}^{2^i - 1} u$ and then adding $2^k - 2^i$ zero columns. For each $i \in \{0, \dots, k-1\}$, define the $n \times n$ matrix

$$N^{(i)} = [0 | \cdots | 0 | u | \tilde{A} u | \cdots | \tilde{A}^{2^i - 1} u | 0 | \cdots | 0],$$

obtained by concatenating the vectors $u, \dots, \tilde{A}^{2^i-1}u$, adding 2^i zero columns on the left and $2^k - 2^{i+1}$ zero columns on the right. Observe that, for any $i \in \{0, \dots, k-1\}$, the equality

$$M^{(i+1)} = M^{(i)} + \tilde{A}^{2^i} N^{(i)}$$

holds. Moreover, for any $i \in \{1, \dots, k-1\}$, the matrix \tilde{A}^{2^i} can be obtained by multiplying $\tilde{A}^{2^{i-1}}$ by itself. This enables us to compute the n vectors $\tilde{A}^0 u, \tilde{A} u, \tilde{A}^2 u, \dots, \tilde{A}^{n-1} u$ using only $2k-1$ matrix multiplications. We describe in Figure 4 the implementation of this approach in the congested clique model, which uses $O(kn^{1-2/\omega}) = O(n^{1-2/\omega} \log n)$ rounds.

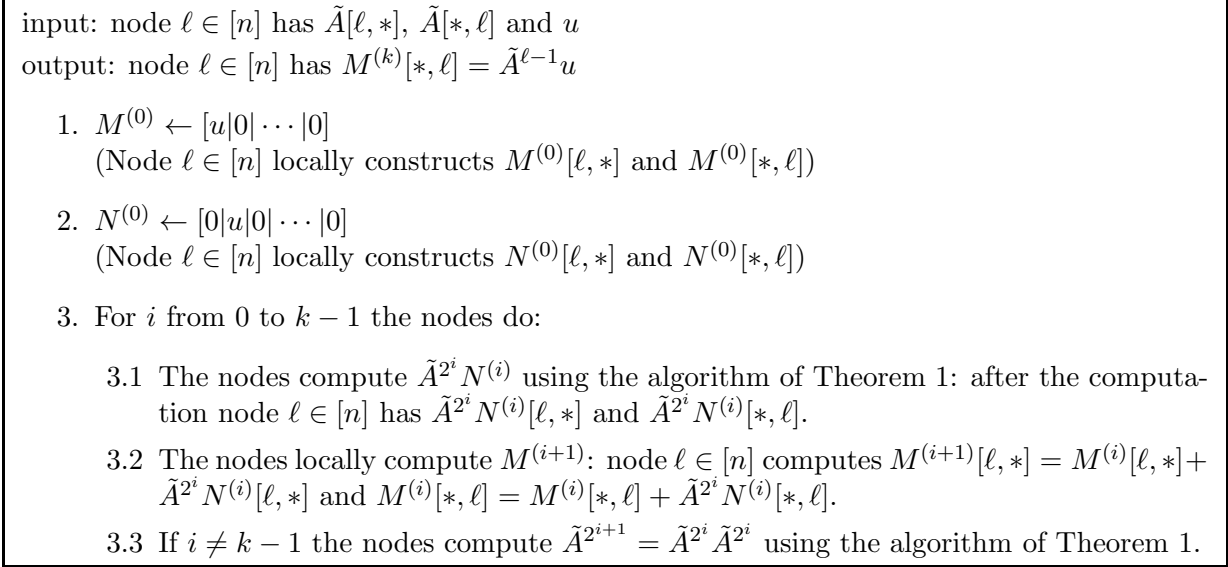


Figure 4: Distributed randomized algorithm for computing the sequence $\tilde{A}^0 u, \tilde{A} u, \tilde{A}^2 u, \dots, \tilde{A}^{n-1} u$ given a matrix $\tilde{A} \in \mathbb{F}^{n \times n}$ and a vector $u \in \mathbb{F}^{n \times 1}$. Here n is a power of two, written $n = 2^k$.

We now describe a $O(n^{1-2/\omega} \log n)$ -round algorithm in the congested clique model that computes, with high probability, the minimal polynomial $\mathbf{minpol}(A)$ of an arbitrary matrix $A \in \mathbb{F}^{n \times n}$. More precisely, the matrix A is initially distributed among the n nodes of the network (node $\ell \in [n]$ receives as input $A[\ell, *]$ and $A[* , \ell]$) and at the end of the computation we would like a designated node of the network (say, node 1) to have the polynomial $\mathbf{minpol}(A)$. The algorithm is as follows. First, a designated node (say, node 1 again) takes two random vectors v, w as in Lemma 2. This node then sends v, w to all the nodes of the network in four rounds of communication using the scheme of Lemma 1. Then the nodes of the network apply the algorithm of Figure 4 with $\tilde{A} = A$ and $u = v$. After this, each node $\ell \in [n]$ owns $A^{\ell-1} v$, and can then compute locally the field element $w A^{\ell-1} v$. All nodes then send their result to node 1 using one round of communication. Node 1 finally computes locally the polynomial $\mathbf{minpol}(A, v, w)$. The complexity of this algorithm is clearly $O(n^{1-2/\omega} \log n)$ rounds. The correctness follows from Lemma 2, which guarantees that $\mathbf{minpol}(A) = \mathbf{minpol}(A, v, w)$ with probability at least $1 - 2n/|\mathbb{F}|$.

Our $O(n^{1-2/\omega} \log n)$ -round algorithm for $\text{INV}(n, \mathbb{F})$ (i.e., for solving the linear system $Ax = b$ where A is invertible) in the congested clique model is as follows. The nodes of the network first use the algorithm of the previous paragraph, so that node 1 obtains with high probability $\mathbf{minpol}(A)$. Let us write $\mathbf{minpol}(A)$ as $m_0 + m_1 \lambda + \dots + m_n \lambda^n$, with $m_n = 1$. From the definition of the

minimal polynomial, we have

$$x = \frac{m_1b + m_2Ab + \dots + m_nA^{n-1}b}{-m_0}. \quad (14)$$

Node 1 then sends the two field elements m_0 and m_ℓ to node ℓ , for each $\ell \in \{1, \dots, n\}$, in two rounds. The nodes of the network use the $O(n^{1-2/\omega} \log n)$ -round algorithm of Figure 4 with $u = b$ and $\tilde{A} = A$, so that each node $\ell \in [n]$ owns $A^{\ell-1}b$ at the end of the computation. Each node $\ell \in [n]$ then locally computes $-\frac{m_\ell}{m_0}A^{\ell-1}b$, and sends the ℓ' -th coordinate of this vector to node ℓ' , for each $\ell' \in [n]$. This can be done in two rounds. Node $\ell \in [n]$ then adds the n elements he receives, which gives $x[\ell]$ from Equation (14).

We now describe our $O(n^{1-2/\omega} \log n)$ -algorithm for $\text{DET}(n, \mathbb{F})$. First, a designated node (say, node 1) takes a random diagonal matrix D as in Lemma 3. This node then sends D to all the nodes of the network using two rounds of communication (using the scheme of Lemma 1). Each node $\ell \in [n]$ of the network then constructs $DA[\ell, *]$ and $DA[*, \ell]$. The nodes of the network then apply the above algorithm computing the minimal polynomial, so that Node 1 obtains with high probability $\mathbf{minpol}(DA)$. Let m_0 denote the constant term of $\mathbf{minpol}(DA)$. Note that the determinant of DA is $(-1)^n m_0$, from Lemma 3, and thus the determinant of A is

$$\frac{(-1)^n m_0}{\prod_{i=1}^n D[i, i]}.$$

Node 1 sends this value all the nodes of the network in one round of communication.

Finally, we present our algorithm for $\text{Rank}(n, \mathbb{F})$. First, we can check with high probability whether $\text{rank}(A) = n$ by computing $\det(A)$ using the algorithm described in the previous paragraph. Therefore we assume below that $\text{rank}(A) < n$, and compute the rank as follows. A designated node (say, node 1) takes three random matrices U, V, D as in Lemma 4, and sends the three matrices to all the nodes of the network in six rounds using the scheme of Lemma 1 (note that each matrix is described by at most n coefficients). The nodes of the network then apply the $O(n^{1-2/\omega})$ -round algorithm of Theorem 1 three times to compute $UAVD$. They then use the $O(n^{1-2/\omega} \log n)$ -round algorithm computing the minimal polynomial, so that Node 1 obtains $\mathbf{minpol}(UAVD)$. Node 1 locally computes $\deg(\mathbf{minpol}(UAVD)) - 1$ and sends this value to all the nodes of the network in one round. The correctness of this algorithm is guaranteed by Lemma 4. \square

6 Applications to Graph-Theoretic Problems

In this section we consider applications of our results to graph-theoretic problems in the congested clique model.

6.1 The All Pair Shortest Paths problem

The All-Pairs Shortest Paths problem (APSP) asks, given a weighted graph $G = (V, E)$, to compute the shortest path between u and v for all pairs of vertices $(u, v) \in V \times V$. For simplicity, but without significant loss of generality, we will assume that the weights are $O(\log n)$ -bit integers. When the graph G is undirected, the definition of the APSP requires the weights to be nonnegative (otherwise there would be negative cycles). When the graph G is directed negative weights are allowed but it is implicitly required that the graph has no non-negative cycle. We say that the graph is unweighted

if the only allowed weight is one. In this subsection the term “adjacency matrix of G ” refers to the $|V| \times |V|$ matrix in which the entry in the i -th row and j -th column is the weight of the edge from the i -th node to the j -th node of the graph if these two nodes are connected and ∞ otherwise.

Let us first describe very briefly the main results concerning the complexity of the APSP in the centralized setting. For undirected unweighted graphs, Seidel [40] showed that the APSP reduces to computing the powers of the adjacency matrix (seen as a matrix over the integers) of the graph, and can thus be solved in $\tilde{O}(n^\omega)$ time. For all other cases, including directed graphs and undirected graphs with arbitrary weights, the standard algebraic way of solving the APSP is to compute the powers of the distance product of the adjacency matrix of the graph. This distance product of an $n \times m$ matrix A by an $m \times n$ matrix B can be trivially solved in time $O(smn^2)$, where s denotes the number of bits needed to represent each entry of A and B , which gives a $\tilde{O}(n^3)$ -time algorithm for the APSP. Despite much research (including recent exciting developments [45]), no significantly better algorithm is known for computing the distance product or solving the general APSP. Faster algorithms for computing the distance product can be nevertheless designed when the entries of A and B are small integers, i.e., the entries are in $\{-M, \dots, -1, 0, 1, \dots, M\} \cup \{\infty\}$ for some integer M . As already mentioned in the proof of Theorem 7, Alon et al. [2] and Takaoka [42] (for the square case) and then Zwick [47] (for the rectangular case) showed that C can be recovered easily from the standard matrix product of the $n \times m$ matrix A' by the $m \times n$ matrix B' where

$$A'[i, j] = \begin{cases} (m+1)^{M-A[i, j]} & \text{if } A[i, j] \neq \infty, \\ 0 & \text{if } A[i, j] = \infty, \end{cases} \quad B'[j, i] = \begin{cases} (m+1)^{M-B[j, i]} & \text{if } B[j, i] \neq \infty, \\ 0 & \text{if } B[j, i] = \infty, \end{cases}$$

for all $(i, j) \in [n] \times [m]$. This implies in particular that the distance product can be computed in $\tilde{O}(Mn^{\omega(\log m / \log n)})$ time, and in particular in $\tilde{O}(Mn^\omega)$ time when $m = n$. Shoshan and Zwick [41] used this technique (for the square case) to obtain a $\tilde{O}(Mn^\omega)$ -time algorithm for the APSP in undirected graphs with weights in $\{0, \dots, M\}$. Zwick used this technique (for the rectangular case) to construct an algorithm for the directed case. He obtained in particular time complexity $O(n^{2.58})$, which has been improved to $O(n^{2.54})$ using the best known upper bound on the exponent of rectangular matrix multiplication [24], when M is constant (and in particular for directed unweighted graphs).

Censor-Hillel et al. [5] showed how to adapt Seidel’s method [40] to solve the APSP over undirected unweighted graphs in $\tilde{O}(n^{1-2/\omega})$ rounds. They also observed that the centralized matrix algorithms for the distance product discussed above can be implemented in the congested clique model, solving $\text{DIST}(n, n, M)$ in $O(\min\{n^{1/3} \log n, Mn^{1-2/\omega}\})$ rounds. While not explicitly stated in [5], this result gives a $O(Mn^{1-2/\omega})$ -round algorithm for the APSP in undirected graphs with weights in $\{0, 1, \dots, M\}$, by observing that the reduction given in [41] from such instances of APSP to the computation of the distance product can be implemented efficiently in the congested clique model. Our improved upper bound (Theorem 4 stated in the introduction) follows directly from our improved algorithm for the computation of the distance product in the congested clique model (Theorem 7).

We now consider the APSP in directed graphs with small integer weights and prove Theorem 5. Besides the new bounds of Theorem 7, we will also use another upper bound on the round complexity of $\text{DIST}(n, m, M)$, which is better for large values of M . This is the bound obtained from Theorem 1 for the choice $\omega(\ell) = 2 + \ell$ corresponding to the implementation of the trivial matrix multiplication algorithm (which works over any semiring, see Section 2) in the congested clique model. We state this bound in the following proposition.

Proposition 5. *The round complexity of $\text{DIST}(n, m, M)$ is*

$$\begin{cases} O(\log M) & \text{if } 0 \leq m \leq \sqrt{n}, \\ O(m^{2/3}n^{-1/3} \log M) & \text{if } m \geq \sqrt{n}. \end{cases}$$

We are now ready to prove Theorem 5.

Proof of Theorem 5. Let A be the adjacency matrix of the graph $G = (V, E)$. The centralized algorithm by Zwick [47] works as follows. The algorithm performs $\lceil \log_{3/2} n \rceil$ iteration, while maintaining an $n \times n$ matrix F initially set to $F = A$. In the k -th iteration, it sets $s = (3/2)^k$, and takes a set $S \subseteq V$ of $O((n \log n)/s)$ vertices chosen uniformly at random from V . The algorithm then constructs the submatrix of size $|S| \times n$ of F , denoted $F[S, *]$, consisting of the rows corresponding to the vertices in S , and the submatrix of size $n \times |S|$ of F , denoted $F[*, S]$, consisting of the columns corresponding to vertices in S . It then puts a cap of sM on the absolute values of the entries of $F[S, *]$ and $F[*, S]$. The last step of the iteration is to compute the distance product $F' = F[*, S] * F[S, *]$ and, for each $(i, j) \in [n \times n]$, replace the (i, j) entry of F by $F'[i, j]$ if $F'[i, j] < F[i, j]$. It can be shown that after the last iteration the entry $F[i, j]$ is the length of the shortest path between the i -th node and the j -th node of the graph, for all $(i, j) \in [n \times n]$, if the graph has no negative cycle.

This centralized algorithm can be implemented easily in the congested clique model. The only part that requires communication between the nodes is the computation of the distance product: at step i the nodes need to compute the distance product of a $n \times m$ matrix by a $m \times n$ matrix with entries of absolute values bounded by $\lceil sM \rceil$ with $m = O((n \log n)/s)$ and $s = (3/2)^i$. We have two strategies to compute the distance product, the algorithm of Proposition 5 and the algorithm of Theorem 7. Since there are only $O(\log n)$ iterations, the total round complexity is

$$\tilde{O} \left(\min \left\{ 1 + (n/s)^{2/3} n^{-1/3}, s + s^{2/\omega(\gamma)} n^{1-2/\omega(\gamma)} \right\} \right), \quad (15)$$

for the value of s that maximizes this expression, where γ denotes the solution of the equation $(1 - \frac{\log s}{\log n})\gamma = 1 - \frac{\log s}{\log n} + (\frac{\log(n/s)}{\log n} - 1)\omega(\gamma)$. As in the analysis for the centralized setting given in [47], the left part of (15) is a decreasing function of s , while the right part is an increasing function of s . Using the best known upper bound on $\omega(\gamma)$ from [24] (see also Figure 5), we can upper bound the total round complexity by $O(n^{0.2096})$.

Note that the above algorithm only computes the lengths of the shortest paths. As in Zwick's centralized algorithm, the shortest paths can be constructed by using exactly the same strategy, but constructing a matrix of witnesses whenever a distance product is computed (as described in Section 3 of [47]), which can be done with the same round complexity. \square

While a result similar to Theorem 5 can be obtained when M is not a constant, by using exactly the same algorithm but keeping the value M in Equation (15), in this case it is complicated to express the result of the numerical optimization in a closed form, so we omit this generalization. An interesting open question is whether the algorithm of Theorem 5 can be derandomized. While Zwick showed that this can be done in the centralized setting by introducing the concept of bridging sets, it does not seem that the algorithm proposed in [47] for constructing bridging sets can be implemented efficiently in the congested clique model.

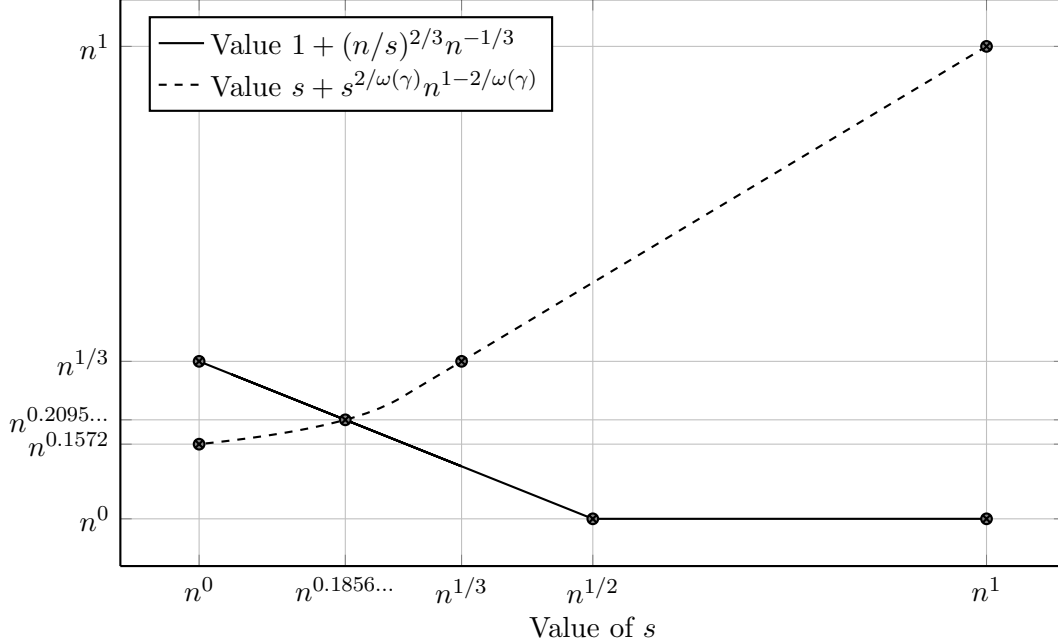


Figure 5: Values of the two parts of Equation (15).

6.2 Diameter computation

Since computing the diameter of a graph trivially reduces to the APSP problem, we immediately obtain the following result.

Corollary 1. *In the congested clique model, the deterministic round complexity of diameter computation in an undirected graph of n vertices with integer weights in $\{0, \dots, M\}$, where M is an integer such that $M \leq n$, is $\tilde{O}(M^{2/\omega} n^{1-2/\omega})$.*

The same reduction can be used to obtain a randomized algorithm for computing the diameter of directed graphs with constant integer weights in $O(n^{0.2096})$ rounds, via Theorem 5. This round complexity can nevertheless be improved. In the centralized setting it is known that over directed graphs with integer weights in $\{-M, \dots, M\}$, but without cycles of negative weights, the diameter can be computed in $O(Mn^\omega)$ time, i.e., faster than the best known centralized algorithm for the corresponding APSP problem. This is a folklore result based on the reductions to the distance product developed in [41, 47] (see also [10]). A close inspection of this approach shows that it can be implemented efficiently in the congested clique model, and Theorem 7 thus implies that the diameter can be computed in $\tilde{O}(M^{2/\omega} n^{1-2/\omega})$ rounds in direct graphs (without cycles of negative weights) with integer weights in $\{-M, \dots, M\}$ as well.

6.3 Maximum matchings

Let $G = (V, E)$ be a simple graph (i.e., an undirected and unweighted graph with no loops or multiple edges), and write $V = \{v_1, \dots, v_n\}$. The Tutte matrix of G is the $n \times n$ symbolic matrix

A such that

$$A[i, j] = \begin{cases} x_{ij} & \text{if } i > j \text{ and } \{v_i, v_j\} \in E, \\ -x_{ij} & \text{if } i < j \text{ and } \{v_i, v_j\} \in E, \\ 0 & \text{otherwise,} \end{cases}$$

for any $(i, j) \in [n] \times [n]$. Let $\nu(G)$ denote the number of edges in a maximum matching of G . Lovász [31] showed that $\text{rank}(A) = 2\nu(G)$. Rabin and Vazirani [38] observed that this equality remains true over any field. As mentioned in [38], this gives a simple randomized algorithm for computing $\text{rank}(A)$, and thus $\nu(G)$, based on Schwartz-Zippel lemma [39, 46]: take a prime $p = \Theta(n^4)$, substitute the variables x_{ij} by random elements from the finite field \mathbb{Z}_p to obtain a matrix \hat{A} over \mathbb{Z}_p , and compute the rank of \hat{A} over \mathbb{Z}_p (which will be equal to $\text{rank}(A)$ with high probability). This approach can clearly be implemented efficiently in the congested clique mode using the rank algorithm of Theorem 3, giving the following result.

Theorem 8. *The randomized round complexity of computing the number of edges in a maximum matching of a simple graph is $O(n^{1-2/\omega} \log n)$.*

Suppose that the graph G has a perfect matching (i.e., n is even and $\nu(G) = n/2$). We say that an edge of G is allowed if it is contained in a least one perfect matching. Rabin and Vazirani [38] further showed that with high probability the following property holds for all edges $\{i, j\}$ of G : the edge $\{i, j\}$ is allowed if and only if $\hat{A}^{-1}[i, j] \neq 0$. The set of allowed edges can thus be obtained from the inverse of the matrix \hat{A} , which can be done in $O(n^{1-1/\omega})$ rounds in the congested clique model using the algorithm of Theorem 2.

An interesting open question is whether finding a perfect matching can be done with the same complexity in the congested clique model. While the best centralized algorithms can find a maximum matching with essentially the same complexity as matrix multiplication [33], they are based on sequential variants of the Gaussian decomposition (e.g., computation of the LUP decomposition) that do not appear to be implementable in parallel.

6.4 Computing the Gallai-Edmonds decomposition of a graph

Let $G = (V, E)$ be a simple graph. We say that a vertex $v \in V$ is critical if it appears in at least one maximum matching, otherwise we say that v is non-critical. Let $D(G) \subseteq V$ denote the set of non-critical vertices, $K(G)$ be the set of vertices in $V \setminus D(G)$ that are adjacent to vertices in $D(G)$, and define $C(G) = V \setminus (D(G) \cup K(G))$. Gallai [15] and Edmonds [14] showed that the triple $(D(G), K(G), C(G))$, called the Gallai-Edmonds decomposition of the graph, gives fundamental information about the structure of the graph, and in particular about its matchings (see, e.g., [32] for a detailed presentation of this theorem). Cheriyan [7] presented efficient algorithms for computing the Gallai-Edmonds decomposition. We adapt this algorithm to the congested clique model to obtain the following result.

Theorem 9. *The randomized round complexity of computing the Gallai-Edmonds decomposition of a simple graph is $O(n^{1-1/\omega})$.*

Proof. We will show how all the nodes of the network can obtain the set of non-critical vertices $D(G)$ in $\tilde{O}(n^{1-1/\omega})$ rounds. Note that $K(G)$ and $C(G)$ can then be computed (and distributed to all the nodes) in a constant number of rounds using the scheme of Lemma 1.

The strategy used in [7] to compute the set of non-critical vertices $D(G)$ in the centralized setting is as follows. Take a prime $p = \Theta(n^3)$ and substitute the variables x_{ij} in the Tutte matrix

of G by random elements from the finite field $\mathbb{F} = \mathbb{Z}_p$. Let \hat{A} denote the matrix obtained. For any $i \in [n]$, let $e_i \in \mathbb{F}^{1 \times n}$ be the row vector with coordinate 1 at position i and coordinate zero at all other positions. Cheriyan [7] showed that with large probability the following property holds for all $i \in [n]$: the vertex v_i is non-critical if and only if

$$\text{rank} \left(\begin{bmatrix} \hat{A} \\ e_i \end{bmatrix} \right) > \text{rank}(\hat{A}). \quad (16)$$

Here $\begin{bmatrix} \hat{A} \\ e_i \end{bmatrix}$ denotes the $(n+1) \times n$ matrix obtained by appending the row e_i at the bottom of \hat{A} . The set $D(G)$ can be computed by checking if Equation (16) holds for each $i \in [n]$, but this is not efficient enough. Observe that Equation (16) holds if and only if e_i is not in the subspace S spanned by the row vectors of \hat{A} . Let $M \in \mathbb{F}^{n \times (n - \text{rank}(\hat{A}))}$ be the matrix representation of a basis for the right null space of \hat{A} (i.e., each column of M is a basis vector of the vector space $S^\perp = \{y \in \mathbb{F}^n \mid \hat{A}y = 0\}$). Observe that for any row vector u we have $uM = 0$ if and only if $u \in (S^\perp)^\perp = S$. Equation (16) then holds if and only if $e_iM \neq 0$. In order to compute e_iM for all $i \in [n]$, we simply need to compute the product of the $n \times n$ identity matrix I_n by the matrix \hat{A} , and check which rows of the product contain at least one non-zero entry.

In the congested clique model this strategy can be implemented in $O(n^{1-2/\omega})$ rounds assuming that the matrix M is available (and distributed among the nodes). We now explain how to construct this matrix using the ideas from [21, 23]. Let r denote the rank of A . Let U, V be two triangular random Toeplitz matrices as in Lemma 4, and write $N = U\hat{A}V$. Decompose N as follows:

$$N = \begin{pmatrix} N_{11} & N_{12} \\ N_{21} & N_{22} \end{pmatrix},$$

where $N_{11} \in \mathbb{F}^{r \times r}$, $N_{12} \in \mathbb{F}^{r \times (n-r)}$, $N_{21} \in \mathbb{F}^{(n-r) \times r}$ and $N_{22} \in \mathbb{F}^{(n-r) \times (n-r)}$. Theorem 2 from [23] and the analysis of Section 5 in [21] shows that, with probability at least $1 - r(r+1)/|\mathbb{F}|$ on the choice of U and V , the submatrix N_{11} is non-singular and the columns of the $n \times (n-r)$ matrix

$$M = V \begin{pmatrix} I_r & -N_{11}^{-1}N_{12} \\ 0_{(n-r) \times r} & I_{n-r} \end{pmatrix} \begin{pmatrix} 0^{r \times (n-r)} \\ I_{n-r} \end{pmatrix}.$$

form a basis of the right null space of \hat{A} . The matrix M can thus be computed with high probability in $O(n^{1-1/\omega})$ rounds by using the algorithm of Theorem 2 for computing the inverse and the algorithm for matrix multiplication of Section 3. \square

As already mentioned, the Gallai-Edmonds decomposition has many applications. In particular, as pointed out by Cheriyan [7], an algorithm computing the Gallai-Edmonds decomposition immediately yields an algorithm computing a minimum vertex cover in a bipartite graph. Cheriyan also presented other graph-theoretic problems that can be solved using variants of his approach for computing the Gallai-Edmonds decomposition: finding the canonical partition of an elementary graph, computing the maximum number of vertex disjoint paths between two subsets of vertices of a graph, and computing a minimal separator between two subsets of vertices of a graph. A close inspection of these variants (Sections 3.3 and 4 in [7]) shows that they can also be implemented efficiently in the congested clique model, giving again algorithms with round complexity $O(n^{1-1/\omega})$.

Acknowledgments

The author is grateful to Arne Storjohann for precious help concerning the computation of the determinant and to anonymous reviewers for their comments. This work is supported by the Grant-in-Aid for Young Scientists (A) No. 16H05853, the Grant-in-Aid for Scientific Research (A) No. 16H01705, and the Grant-in-Aid for Scientific Research on Innovative Areas No. 24106009 of the Japan Society for the Promotion of Science and the Ministry of Education, Culture, Sports, Science and Technology in Japan.

References

- [1] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley Publishing Company, 1974.
- [2] Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997.
- [3] James R. Bunch and John E. Hopcroft. Factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974.
- [4] Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. *Algebraic complexity theory*. Springer, 1997.
- [5] Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. In *Proceedings of the 34th Symposium on Principles of Distributed Computing*, pages 143–152, 2015.
- [6] Li Chen, Wayne Eberly, Erich Kaltofen, B. David Saunders, William J. Turner, and Gilles Villard. Efficient matrix preconditioners for black box linear algebra. *Linear Algebra and its Applications*, 343-344:119–146, 2002.
- [7] Joseph Cheriyan. Randomized $\tilde{O}(M(|V|))$ algorithms for problems in matching theory. *SIAM Journal on Computing*, 26(6):1635–1669, 1997.
- [8] Jeff I. Chu and Georg Schnitger. Communication complexity of matrix computation over finite fields. *Mathematical Systems Theory*, 28(3):215–228, 1995.
- [9] Laszlo Csanky. Fast parallel matrix inversion algorithms. In *Proceedings of the 16th Annual Symposium on Foundations of Computer Science*, pages 11–12, 1975.
- [10] Marek Cygan, Harold N. Gabow, and Piotr Sankowski. Algorithmic applications of Baur-Strassen’s theorem: Shortest cycles, diameter and matchings. In *53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 531–540, 2012.
- [11] Anindya De, Piyush P. Kurur, Chandan Saha, and Ramprasad Saptharishi. Fast integer multiplication using modular arithmetic. *SIAM Journal on Computing*, 42(2):685–699, 2013.
- [12] Danny Dolev, Christoph Lenzen, and Shir Peled. “Tri, Tri Again”: Finding triangles and small subgraphs in a distributed setting - (extended abstract). In *Proceedings of the 26th International Symposium on Distributed Computing*, pages 195–209, 2012.

- [13] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 367–376, 2014.
- [14] Jack Edmonds. Paths, tree and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
- [15] Tibor Gallai. Maximale Systeme unabhängiger Kanten. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 9:401–413, 1964.
- [16] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, 2003.
- [17] James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, and Michele Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and MST. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 91–100, 2015.
- [18] James W. Hegeman and Sriram V. Pemmaraju. Lessons from the congested clique applied to MapReduce. In *Proceedings of the 21st International Colloquium on Structural Information and Communication Complexity*, pages 149–164, 2014.
- [19] James W. Hegeman, Sriram V. Pemmaraju, and Vivek Sardeshmukh. Near-constant-time distributed algorithms on a congested clique. In *Proceedings of the 28th International Symposium on Distributed Computing*, pages 514–530, 2014.
- [20] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, pages 489–498, 2016.
- [21] Erich Kaltofen and Victor Y. Pan. Processor efficient parallel solution of linear systems over an abstract field. In *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 180–191, 1991.
- [22] Erich Kaltofen and Victor Y. Pan. Processor-efficient parallel solution of linear systems II: the positive characteristic and singular cases (extended abstract). In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 714–723, 1992.
- [23] Erich Kaltofen and B. David Saunders. On Wiedemann’s method of solving sparse linear systems. In *Proceedings of the 9th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, pages 29–38, 1991.
- [24] François Le Gall. Faster algorithms for rectangular matrix multiplication. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 514–523, 2012.
- [25] François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, pages 296–303, 2014.
- [26] Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 42–50, 2013.

- [27] Christoph Lenzen and Roger Wattenhofer. Tight bounds for parallel randomized load balancing: extended abstract. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 11–20, 2011.
- [28] Yuri Vladimirovich Linnik. On the least prime in an arithmetic progression I. The basic theorem. *Rec. Math. [Mat. Sbornik] N.S.*, 15(57):139 – 178, 1944.
- [29] Yuri Vladimirovich Linnik. On the least prime in an arithmetic progression II. The Deuring-Heilbronn phenomenon. *Rec. Math. [Mat. Sbornik] N.S.*, 15(57):347 – 368, 1944.
- [30] Zvi Lotker, Elan Pavlov, Boaz Patt-Shamir, and David Peleg. MST construction in $o(\log \log n)$ communication rounds. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 94–100, 2003.
- [31] László Lovász. On determinants, matchings, and random algorithms. In *Fundamentals of Computation Theory*, pages 565–574, 1979.
- [32] László Lovász and Michael D. Plummer. *Matching Theory*. American Mathematical Society, 2009.
- [33] Marcin Mucha and Piotr Sankowski. Maximum matchings via Gaussian elimination. In *Proceedings of the 45th Symposium on Foundations of Computer Science*, pages 248–255, 2004.
- [34] Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Proceedings of the 46th Symposium on Theory of Computing*, pages 565–573, 2014.
- [35] Boaz Patt-Shamir and Marat Teplitsky. The round complexity of distributed sorting: extended abstract. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing*, pages 249–256, 2011.
- [36] David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, 2000.
- [37] Franco P. Preparata and Dilip V. Sarwate. An improved parallel processor bound in fast matrix inversion. *Information Processing Letters*, 7(3):148–150, 1978.
- [38] Michael O. Rabin and Vijay V. Vazirani. Maximum matchings in general graphs through randomization. *Journal of Algorithms*, 10(4):557–567, 1989.
- [39] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980.
- [40] Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of Computer and System Sciences*, 51(3):400–403, 1995.
- [41] Avi Shoshan and Uri Zwick. All pairs shortest paths in undirected graphs with integer weights. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 605–615, 1999.
- [42] Tadao Takaoka. Subcubic cost algorithms for the all pairs shortest path problem. *Algorithmica*, 20(3):309–318, 1998.

- [43] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Symposium on Theory of Computing*, pages 887–898, 2012.
- [44] Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, 1986.
- [45] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the 46th Symposium on Theory of Computing*, pages 664–673, 2014.
- [46] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 216–226, 1979.
- [47] Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM*, 49(3):289–317, 2002.